DeepAnT: A Deep Learning Approach for Unsupervised Anomaly Detection in Time Series

Bastien Batardiere, Omar Souaidi

Ecole Normale Supérieure Paris-Saclay

bastien.batardiere@ens-paris-saclay.fr, omar.souaidi@student.ecp.fr

Abstract

The problem of anomaly detection on time series is to predict whether a newly observed time series is normal or abnormal. It is very useful in many monitoring applications such as video surveillance and signal recognition.We present here DeepAnt [1], an unsupervised approach based on convolutionnal neural networks. We compare our approach against other anomaly detection systems on real data.

1. Introduction

As stated in [2], anomaly detection is the identification of rare items, events or observations which raise suspicions by differing significantly from the majority of the data. DeepAnT is an unsupervised approach to this problem that aims at capturing the distribution of the data in order to extract the anomalies. It uses CNN as forecasting module. It detects anomalies by comparing the actual data point to its forecasting. Here the method is unsupervised since we are using the actual Time Serie as target.

2. Paper

2.1. Method

The approach consists of two steps :

- Make a prediction
- Given the prediction, classifying each data point as normal or abnormal

2.2. Predictor

2.2.1. Convolutionnal Neural Network

A convolutional neural network (CNN) is a multi-layer artificial neural network that has been successful recognizing visual patterns. Conventionally, A CNN consists of alternate layers of convolutional layers and sub-sampling layers on the bottom and several fully-connected layers following them. A Convolutional layers usually consits in three steps :

- Convolution operation
- Non-linear Activation function
- Pooling layer

The convolution operation is here to smooth and extract the best parameters of the given input. The activation allows the to compute nontrivial problems. Finally, pooling layer is here to reduce dimensionnality by summarizing the presence of features in patches of the feature map.

CNN are very efficient thanks to their shared parameters that reduces the memory burden.

2.2.2. CNN for time serie prediction

Consider a time series :

$$\{x_0, x_1, \dots x_n\}$$

Here we consider a uni-variant time serie but it can be applied to multi-variant time serie straightforwardly. We aim at predicting a datapoint givent the previous data points. More specifically, given a window size w and a prediction window p_w, we will predict $(x_N, \dots x_{N+p_w-1})$ from $(x_{N-w}, \dots, x_{N-1})$. $(x_N, \dots, x_{N+p_w-1})$ is the label, that is why the approach is unsupervised, we use the time serie as label.

We use CNN as predictor. Denote ϕ a neural network, we will want this quantity to be small.

$$\frac{1}{n}\sum_{N=w}^{n-p_w+1} l(\phi(x_{N-w}, \dots, x_{N-1}) - (x_N, \dots x_{N+p_w-1}))$$

where l is a loss we want to minimize. The neural network will update the parameters of its layers according to this loss with a gradient descent algorithm.

2.2.3. Architecture summary

The architecture is quite simple. It includes two convolutionnal layer, followed by one fully-connected layer, as describes in Figure 1. The output dimension is equal to p_w.

2.3. Anomaly detector

Once we have a prediction for the next p_w time stamps, we need to classify it as normal or abnormal. Naturally, we will classify as abnormal if the actual value is too far from our prediction. This implies the need of a threshold α.

$$(x_N, .., x_{N+p_w-1}) \text{ is abnormal}$$

$$\Updownarrow$$

$$d((x_N, .., x_{N+p_w-1}), \phi(x_{N-w}, .., x_{N-1})) > \alpha$$

where d is a distance. Typically, we will take the euclidean distance. The threshold needs to be tuned, and it can vary from the context. Indeed, some applications will prefer to detect every value that looks a little bit abnormal, whereas some will prefer to focus on the most obvious ones.

2.4. Training

We need to train our model in order to optimize the parameters. However, it would not be consistent to predict anomaly on data you have already seen, because the model would overfit. That is why we split the dataset in two parts : training and testing. We

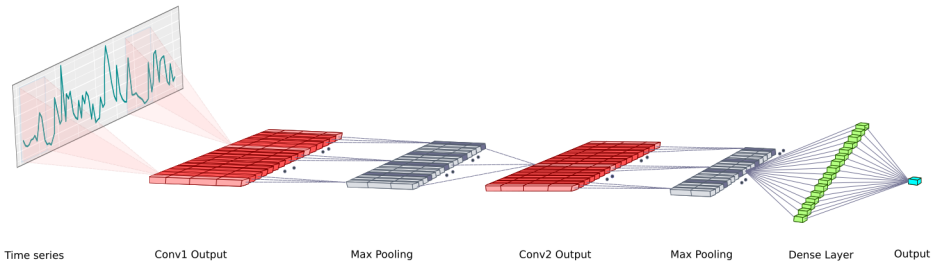


Figure 1: *Architecture*

take the first 40 pourcent as training, for optimization, and we can detect anomaly on the second part. Note that we can't train the model on any data. Indeed, if the dataset allready has too much anomalies, the model won't be able to learn properly. The author of DeepAnT recommends that the anomaly rate should be lower than 5%.

3. Implementation

We implemented the paper to have a clear understanding of how it works. The code can be found at https://github.com/omarsou/DeepAnt_timeseries_anomaly . We will discuss here the main steps of the implementation. We first took random time series on the internet in order to check the implementation, since the most difficult part is the prediction one. The dataset can be found here : https://github.com/matrix0415/DeepLearning-Examples/blob/master/dataset/uci-sml/NEW-DATA-1.T15.txt .

3.1. Preprocessing

Before implementing a neural network, you first need to process the data. To begin, we will replace the NaN values with the mean value. Note that this step is important for our task, since too many NaN values will completely corrupt the training, as stated before (NaN values can be considered as anomaly values). Next we need to modify the data so that the neural network can take as input a sequence of length w, and focus on the label of size p_w. We also need to split the dataset, the training and testing one.

This processing is done by the function *make_interpretable*.

3.2. Convolutionnal Neural Network

We used the package PyTorch [3] to implement the neural Network. We first need to define the layers one by one and the *forward* function so that the input can go through the network. We put 2 convolutionnal layers, and 2 fully connected layer. It is one more than the paper recommendation, but we found better results with this setting. Next, we have to train the network on the training set. We used the method *fit*. We optimized the neural network parameter using Stochastic Gradient Descent. We used the Mean Squared Error loss. We set the learning rate to 0.01 and decreased it by a factor of 0.8 when the loss began to plateau. We used a learning rate scheduler for this task.

The network is now trained, but we will have to tune it a little bit when we will have labels for anomalies to compare with. For instance, we don't know if we should add a dropout layer. Indeed, without dropout, we can train the network for a thousand epochs and the loss will still decrease. However, it may overfit. We may also change the loss function.

The overfitting is double edged here. First, it does not generalise well to other data, as we can guess. Moreover, if some data points are abnormals, the network will try to learn those abnormal datapoints. In Figure 3, we can guess some ambiguous datapoints, given the rest of the time serie. Even if we have a low percentage of abnormal points, forcing the network to learn those is not efficient. An early stopping may solve this problem.

3.3. Anomaly detection

The anomaly detection is quite straightforward, since we only need to compare our prediction to the actual value of the time serie. This is done by the function *predict_error*. We have to set a threshold to classify the data as normal or abnormal. This threshold is very task-dependant and need to be tune. We used here the euclidean distance. Note that the choice of the distance is not very important.

3.4. Non-applicable datasets

The paper recommends a percentage of abnormal points smaller than 5%. Indeed, we see here the result of a dataset which has a big number of NaN values, which make the training hopeless.

4. Anomaly Detection Experimentation

We looked for labeled dataset; for instance, we contacted Yahoo in order to get their dataset [1], but they answered us quite late, thus we decided to go with the NAB (Numenta Anomaly Benchmark) Dataset [4]. We considered two data streams : New York City Taxi Trafic and the Machine Temperature System Failure. It is hard to use these datasets as practical anomaly detection benchmark since Each data point with ground truth anomaly label is centered by a defined anomaly window (10% the length of a data file), which makes the ground truth label of normal data points also as anomalous. For example, for an anomaly window of size 350, all of the 350 data points in a data stream are labeled anomalous, whereas there are just 2-3 actual anomalies in the middle of this anomaly window. Hence, we will use it just to do our experimentation, the metrics we are going to compute will only serve as a comparison between differents approaches.

4.1. Preprocessing

First of all, in order to have less anomaly points, we decided to resample our time series data. Concerning the NYC Taxi Dataset, we downsample by decreasing the frequency from 30 minutes bins to 12 hours bins and sumed the values of the times-

[1] https://webscope.sandbox.yahoo.com/catalog.php?datatype=sdid=70

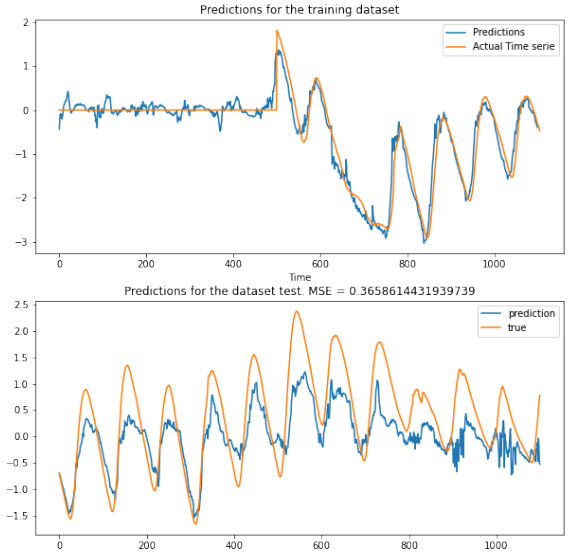


Figure 2: *Predictions for a non-applicable dataset*

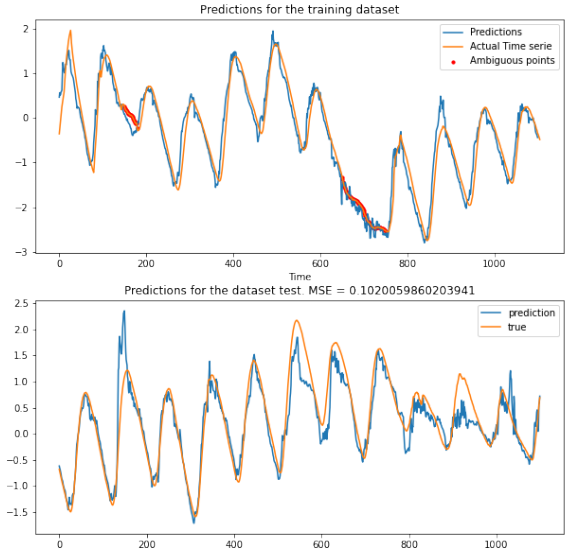
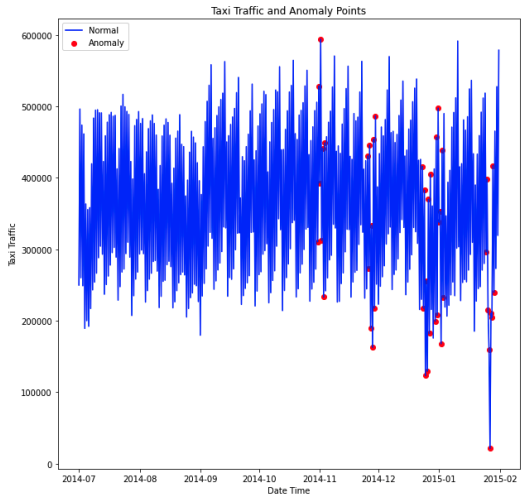
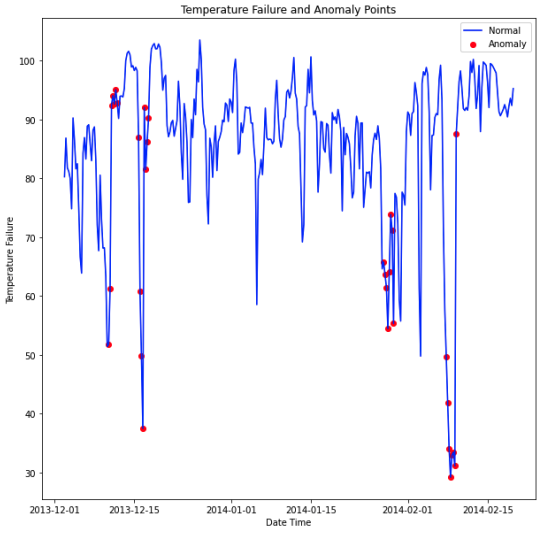


Figure 3: *Predictions for the training and testing dataset*



(a) *New York City Taxi Trafic*



(b) *Temperature Failure System*

Figure 4: *Visualization of the time series dataset and the anomaly points.*

tamps falling into a bin. Concerning the Temperature Dataset, we also downsample by decreasing the frequency from 5 minutes bins to 5 hours bins and averaged the values of the timestamps falling into a bin. You can see in figure 4 the evolution in time of each of the time series and the anomaly points in red.

4.2. Evaluation Metric

We have used F1-score for this detailed evaluation so that an overall performance of an algorithm can be reported.

$$F1 = 2\frac{precisionxrecall}{precision + recall} \tag{1}$$

4.3. Experimental Setup

We wanted to compare the DeepAnT approach with other algorithms. We used 8 differents methods (including two methods based on DeepAnT). The first one is a basic anomaly detection method based on statistics (**Hotelling's T2 test**), the sec-

ond one is an Unsupervised kernel-based anomaly detection method (**One-Class SVM**) [5], the third one is an Unsupervised tree-based anomaly detection method (**Isolation Forest**) [6], the fourth one is Unsupervised density-based anomaly detection method, which measures the local deviation of density of a given sample with respect to its neighbors (**Local Outlier Factor**) [7]. For the four last approaches, we will give more details.

4.3.1. Fb Prophet

The fifth approach is based on FbProphet[2]. Prophet is a procedure for forecasting time series data based on an additive model where non-linear trends are fit with yearly, weekly, and daily seasonality, plus holiday effects. In this approach, we use all the data and fed it to the fbprophet model. By default Prophet will return uncertainty intervals for the forecast. First we initialize the prophet model with 95% confidence (which means that we only let 5% margin of error, it is just the confidence level fixed ofr any statistical analysis). Now for each y^t_{pred} that the model will predict at a timestep t, it will also output the interval $[y^t_{lower}, y^t_{upper}]$ in which we are 95% confident that our predicted values will fall. Thus, a point at timestep t will be considered as an anomaly if :

$$|y^t_{pred} - y^t_{true}| >= \frac{y^t_{upper} - y^t_{lower}}{2} \tag{2}$$

4.3.2. Variance Based Method

This is variance based method with assumption of the normal distribution against the data. Let denote the set of values of the time series y such that $y = [y_0, y_1, ..., y_T]$. Let $\bar{y} = \frac{1}{T}\sum_{t=0}^{T} y_t$ and $\sigma(y) = \frac{1}{T-1}\sum_{t=0}^{T}(y_t - \bar{y})^2$ and $thres_{up} = \bar{y} + 1.5\sigma(y)$ and $thres_{low} = \bar{y} - 1.5\sigma(y)$. Hence a point at timestep t will be considered as an anomaly if $y_t > thres_{up}$ or $y_t < thres_{low}$

4.3.3. DeepAnT Method

We will propose two different approaches but the principle remains the same. We will fit a model on the dataset y and we will predict the same time series y_{pred}, we will then compute the absolute difference at each timestep $t : \delta_t = |y^t_{true} - y^t_{pred}|$, hence we will obtain a vector $\delta = [\delta_0, \delta_1, ..., \delta_T]$. In order to set a threshold to distinguish between abnormal point and normal point, we will assume (which is true) that the dataset contains only 10% of anomaly points, hence we will consider that $threshold = \delta_k$ where k correspond to the indice of the smallest δ_i among the 10% percent biggest δ.

The first approach consists of training all the dataset with the DeepAnT Framework and predict at each timestep t the value with the model.

The second approach (OurDeepAnT) can be used to overcome the overfitting, for that, we can split our data into five different split (train/test), and we train five DeepAnT Models on each train of these five splits. As each test set of these splits are disjoint and that their union forms the whole timeseries, we will predict all the values.

4.4. Results

Table 1 shows the result of each algorithm on both Datasets. The FbProphet approach seems to be the most effective, which is understandable since we are using robust statisticals tools that are combined with the time series modelisation. However, we can see that the approach that we propose using the DeepAnT Model seems to be more effective.

As we can see in figure 5, 6 and 7, our approach alongside the fbprophet approach and the variancebased method do not output very bad anomaly points, which shows that these algorithms are effective.

[2]https://github.com/facebook/prophet

	NYC Taxi	Machine Temperature Failure
Hotelling	0.16	0.48
One-class SVM	0.20	0.39
Isolation Forest	0.28	0.57
LocalOutlierFactor	0.28	0.57
varianceBased	0.25	**0.61**
FbProphet	**0.43**	0.58
DeepAnT	0.23	0.42
OurDeepAnT	0.35	0.42

Table 1: *Results on both Datasets using F1-Score as Metric*

5. Conclusion

In this project, we presented a deep learning based approach for the detection of anomalies in tme series data. Since the approach is unsupervised, it requires no labels for anomalies. We also implemented from scratch this method and evaluated on 2 differents datasets in comparison with 6 other differents algorithms. We also propose an approach to use DeepAnT to detect anomalies by avoiding the overfitting phenomena that can occur during the training of a neural network.

6. References

[1] M. Munir, S. Siddiqui, A. Dengel, and S. Ahmed, "Deepant: A deep learning approach for unsupervised anomaly detection in time series," *IEEE Access*, vol. 7, pp. 1991–2005, 2019.

[2] A. Zimek and E. Schubert, *Outlier Detection*. New York, NY: Springer New York, 2017, pp. 1–5. [Online]. Available: https://doi.org/10.1007/978-1-4899-7993-3$_8$0719 − 1

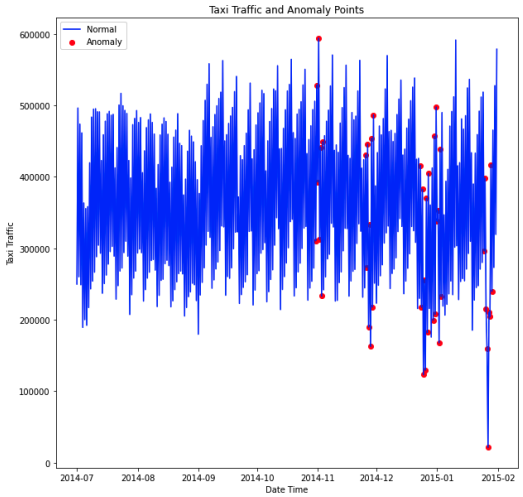
[3] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," pp. 8024–8035, 2019. [Online]. Available: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

[4] A. Lavin and S. Ahmad, "Evaluating real-time anomaly detection algorithms – the numenta anomaly benchmark," *2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA)*, Dec 2015. [Online]. Available: http://dx.doi.org/10.1109/ICMLA.2015.141

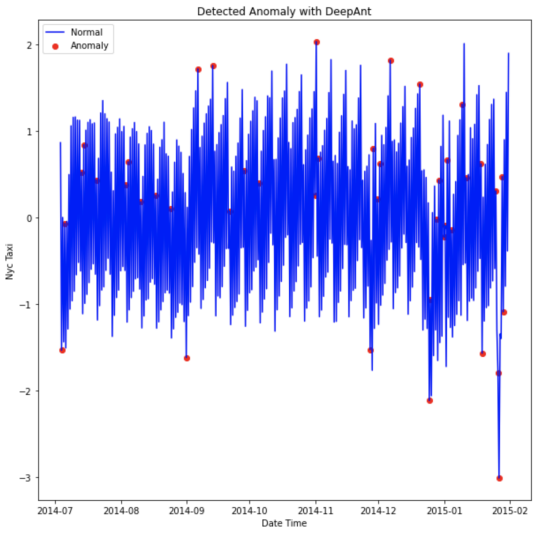
[5] B. Jin, Y. Chen, D. Li, K. Poolla, and A. Sangiovanni-Vincentelli, "A one-class support vector machine calibration method for time series change point detection," 2019.

[6] G. Staerman, P. Mozharovskyi, S. Clémençon, and F. d'Alché Buc, "Functional isolation forest," 2019.

[7] K. Babaei, Z. Chen, and T. Maul, "Detecting point outliers using prune-based outlier factor (plof)," 2019.

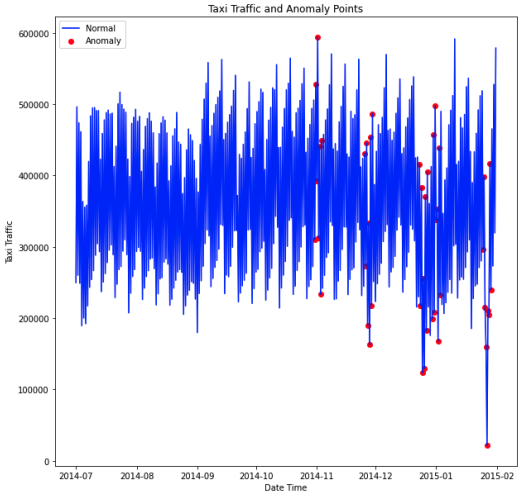


(a) *Ground Truth*

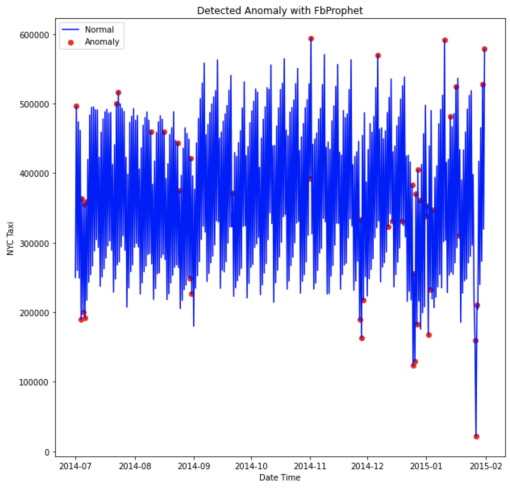


(b) *OurDeepAnT Approach*

Figure 5: *Comparaison between the ground truth Anomaly point and the ones given by OurDeepAnT Approach on the NYC dataset.*

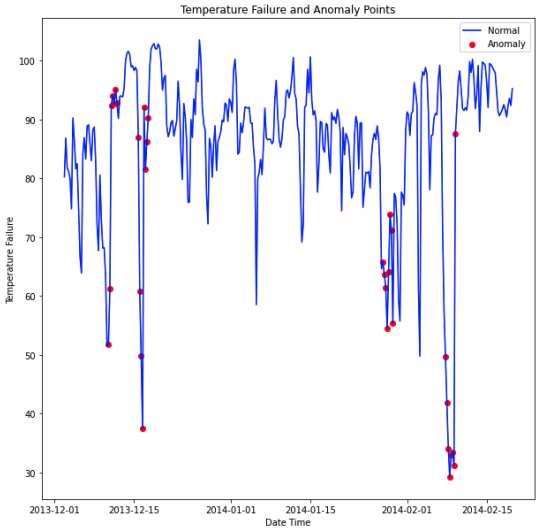


(a) *Ground Truth*

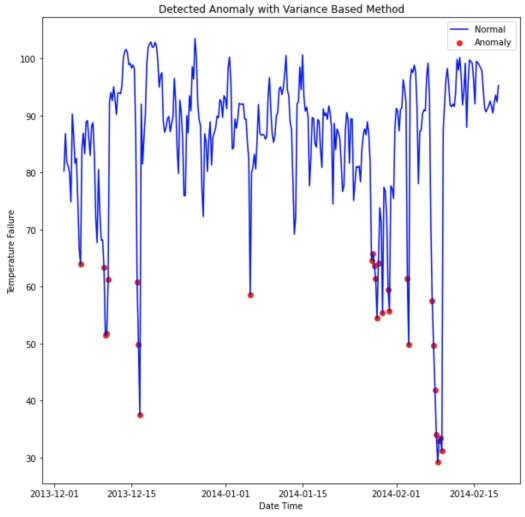


(b) *FbProphet Approach*

Figure 6: *Comparaison between the ground truth Anomaly point and the ones given by FbProphet Approach on the NYC dataset.*



(a) *Ground Truth*



(b) *VarianceBased Approach*

Figure 7: *Comparaison between the ground truth Anomaly point and the ones given by VarianceBased Approach on the Temperature Failure System Dataset.*